QP tries are smaller and faster than crit-bit trees

Tony Finch
University of Cambridge
dot@dotat.at

ABSTRACT

A trie data structure stores an ordered set of keys; the branching structure of a trie depends on the lexical properties of its keys independent of the order of insertion. Compact implementations of PATRICIA binary tries called crit-bit trees [2] have just two words of overhead per item stored.

A hash array mapped trie (HAMT) [1] has wide fan-out, indexing each tree node using several hashed key bits; each node is compressed using the population count of a bitmap to omit NULL child pointers. Bagwell sketches an un-hashed pure trie variant of HAMT in section 5 but doesn't eliminate redundant single-child nodes like crit-bit trees.

Our contribution, QP tries [3], are similar to crit-bit trees but test 5 bits per indirection instead of 1, using the HAMT bitmap POPCNT trick to keep overhead to at most two 64 bit words per item. QP tries prefetch the child pointer array while calculating which child is next; this reduces indirection latency and increases performance by about 5%. QP tries have variable-sized nodes, so stress memory allocation more than crit-bit tries, but are usually much cheaper in other respects.

We created similar implementations of QP tries and crit-bit trees, and benchmarked them using lists of: English words; identifiers in the BIND9 source code; domain names from a university; Alexa top million domain names. We measured average: trie depth; space overhead per item; mutation and search time.

BODY

Typical QP trie
depth is 0.35-0.40
space is 0.5-0.6
time is 0.6-0.8
of equivalent crit-bit tree.

REFERENCES

[1] P. Bagwell. Ideal hash trees. LAMP-REPORT 2001-001, École polytechnique fédérale de Lausanne EPFL.

[2] D. J. Bernstein. Crit-bit trees, 2004. `http://cr.yp.to/critbit.html`.

[3] T. Finch. QP tries, Oct. 2015. `http://dotat.at/prog/qp/`.

Volume 4 of Tiny Transactions on Computer Science